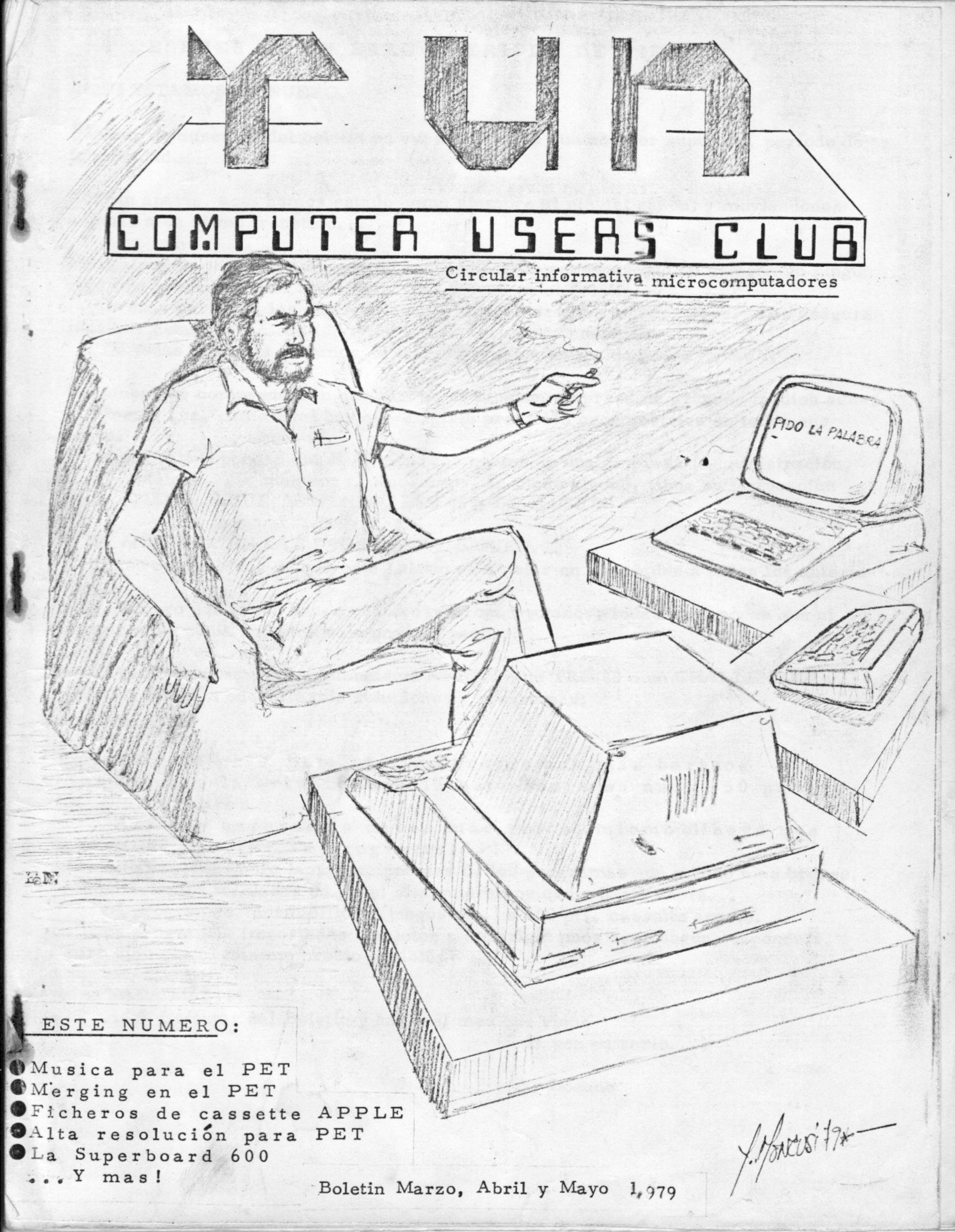


CUC

COMPUTER USERS CLUB

Circular informativa microcomputadores

ESTE NUMERO:

Boletin Marzo, Abril y Mayo 1,979

Y DESPUES DE UN LARGO PERIODO DE TIEMPO...

AQUI ESTAMOS DE NUEVO.

Periodo de ausencia del boletín en vuestras casas que no, por supuesto, período de inactividad.

Boletín aparte, aquí hemos estado como siempre al pié del cañón; y moviendonos además a un ritmo frenético...

Naturalmente creemos que existen entre vosotros personas que no estarán en ningun modo de acuerdo con el retraso de tres meses en el boletín.
A esas personas les suplicamos que no sean crueles con nosotros. Les aseguramos que estamos haciendo MAS de lo que realmente podemos.
De todas maneras creemos en la comprension de los usuarios de RUN.

Creemos que comprenderan que Run no es ninguna empresa de grandes medios económicos y que, como a tal, hay que exigirle precisión cronométrica en todos sus actos.
Es significativo pensar que RUN tiene los gastos de una empresa (Administración, local, teléfono, [illegible] máquinas, etc.) pero, en modo alguno, tiene su facturación
PERO TIENE, SIGUE TENIENDO, ESE 95% DE ILUSION

Y LA MUESTRA EN LAS PAGINAS QUE SIGUEN...
Estamos convencidos de que este boletín va a anular en novedades a todos los anteriores...
Ved lo nuevo que puede hacer vuestro microcomputador, todas esas cosas que ni imaginar se podía hace tan solo unas semanas.

Y también para los que teneis un APPLE o un TRS-80 o un CHALLENGER...
En RUN también encontrareis soluciones y programas.

MAS...:
A partir de este momento los programas más baratos
Resulta que la actual libreria está ahora en unos 350 programas diferentes.
Asi que esperamos daros mas por el mismo dinero que pensabais gastar en programas!!!
Por ello tambien ahora las descripciones de los programas son mucho mas breves, necesitariamos toneladas de papel si siguieramos como hasta ahora...
Los programas "normalillos" (juegos etc.) a 500 pts. cassette aparte.
Los programas importados o sujetos a copyright pues eso, segun, depende...
Pero siempre al mínimo precio posible!!

... NADA MAS!!!
...A disfrutar del Boletín y hasta el mes que viene
(Esta vez en serio...)

A. Lozano

INTERPRETACION DE LOS LISTADOS DE ASSEMBLER
... Y LA MANERA DE INSERTARLOS EN LA MEMORIA USANDO EL BASIC

He aquí un ejemplo tipico del contenido de un segmento de memoria (en hexa) cuando dicho segmento esta cargado con programa OS (código máquina):

033A = A9 00 85 53 85 56 38 A5 51 E9 4F 30 03 E6 54.... etc.

Eso quiere decir que la posicion 033A (equivalente a 826 en decimal) está cargada con un A9 ; la posicion siguiente con un 00, la que sigue (033C) con un 85...
Y así sucesivamente hasta el final del programa

Como se da la característica de que en nemónicos del lenguaje de máquina, una instrucción puede necesitar de mas de una posición de memoria para ser definida, es probable que se de el caso de confusión por parte de los principiantes cuando, al pretender entrar en la máquina un programa en OS a partir de un listado de ASSEMBLER, no saben exactamente que códigos, de todos los representados, son los que hay que convertir en decimal y POKARlos en la memoria...

El listado en hexa con que empezamos esta explicación adquiriria la siguiente pinta visto en formato de ASSEMBLER:

```
033A                    ORG      $033A

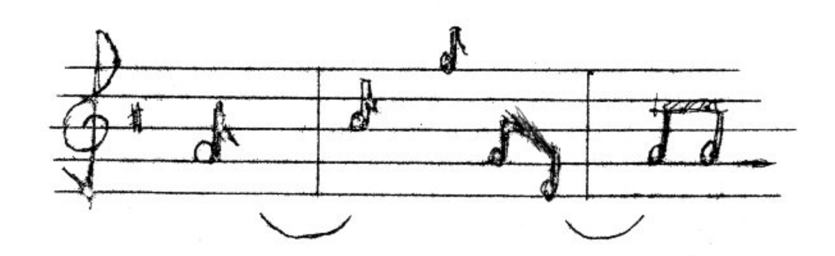
033A  A9 00    START    LDAIM    $00
033C  85 53             STA      $0053
033E  85 56             STA      $0056
0340  38                SEC
0341  A5 51             LDA      $0051
0343  E9 4F             SBCIM    $4F
.
.
..... etcetera...
```

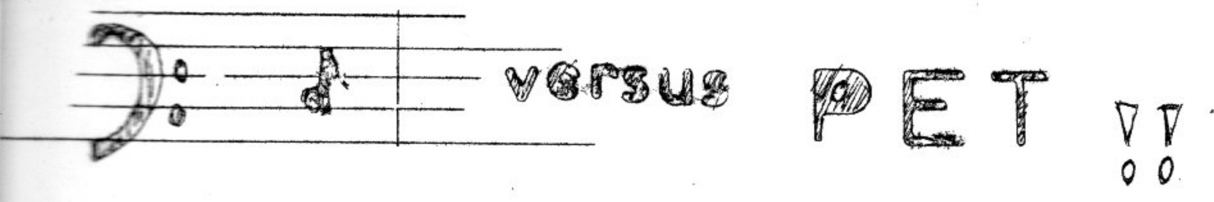
Asi vemos que la zona de interes en el listado de arriba es, precisamente la que corresponde al segundo campo (2º tabulación) que es donde estan los códigos en hexa del programa.

En general, un listado de ASSEMBLER, se descompone de la siguiente manera:
(se cuentan las tabulaciones -campos- comenzando a partir de la izquierda)

ler campo.......dirección de memoria en que comienza la instrucción
2º " intrucción (de uno a tres códigos en hexadecimal)
3er " Etiquetas
4º " Nemónico
5º " Operando de la instrucción

APPLE II...





Sucede que un día tuvimos ocasión de jugar durante un buen rato con un APPLE II.
Sorprendidos nos quedamos cuando pudimos comprobar la enorme capacidad del computador.
Entre sus posibilidades, una de las más diveritdas fue la facilidad que tiene para ejecutar o componer música (a través del altavoz que lleva incorporado)
Inmediatamente pensamos en que era lógico que el computador más caro de los que teníamos en RUN en aquel momento, hiciera funciones más "caras" que los otros.
Pero no nos conformamos y deducimos que si ambos llevan la misma unidad central (6502), evidentemente sus posibilidades deberían ser muy similares, bytes de memoria aparte.
Se trataba de implementar en nuestro PET las funciones en lenguaje máquina que generan los tonos del APPLE (de hecho tuvimos ocasión de observar un programa en el cual el APPLE habla, es decir, sintetiza voces humanas)
Y nos pusimos a la obra.
Comenzamos a desensamblar las rutinas del APPLE; después de un montón de horas de trabajo llegamos a la conclusión de que estábamos perdiendo el tiempo:

! EL PET LLEVA IMPLEMENTADA DE ORIGEN LA POSIBILIDAD DE GENERAR TONOS MUSICALES!

Nada menos que 255 notas diferentes de las que las dos últimas ya llegan a ser inaudibles.
El procedimiento es muy sencillo.

Consiste en hacer trabajar la VIA como divisor de frecuencias del oscilador de cristal.
POKE 59467,16: POKE 59466,15: POKE 59466,100 (RETURN)

Si después de haber ejecutado esta linea conectamos la entrada del amplificador de baja frecuencia a las patas M y N del USERS PORT (N= masa), obtendremos en los altavoces de dicho amplificador de un tono de 612, 7Hz
De aquí a hacerle hacer música no hay más que un paso:
Dejar el amplificador conectado y entrar el programa de las páginas siguientes.

```
1000  REM TOCAR NOTA N DURANTE D JIFFIES
1010  T1 = T1
1020  POKE  AF, P (N)
1030  POKE  AW, W(N)
1050  IF(TI-T)<D THEN 1050
1060  POKE  AW, Ø
1070  RETURN

1100  REN DATOS DE FRECUENCIA
1110  DATA 227, 211, 199, 187, 177, 167
1120  DATA 157, 148, 140, 132, 124, 117

2000  REM INICIALIZACION
2010  DIM  P(36), W(36)
2020  W = 15
2030  FOR J = 1 TO 36
2040  IF J = 13 THEN RESTORE : W = 51
2050  IF J = 25 THEN RESTORE:  W= 85
2060  READ P(J) : W(J) =  W: NEXT J
2070  AW = 59466: AF = 59464
2080  POKE  59467, 16
2090  RETURN
```

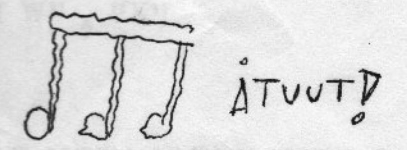
La subrutina 2000 sirve para preparar al PET para hacer música y la 1000 para ejecutar.

EJEMPLO

COSUB 2000 (RETURN) --PREPARA AL PET COMO ORGANO--
N = 1: D = 20: GOSUB 1000 (RETURN)
Dará la nota "do" (1=do, 2=do :: , 3=re . . . etc.) Durante D jiffies.
Se cubren 3 octavas (desde N= 1 a 36)
El mejor ejemplo es el programa de las páginas siguientes.
La imitación de la flauta es perfecta.

NOTA :
Disponemos en RUN de pequeños amplificados enchufables directamente al USERS PORT al precio de 2000,- ptas.
Por si alguién no se atreviera a trastear con su grupo de alta fidelidad y el PET . . .

!OJO! después de haber hecho funcionar el programa musical, el PET se "olvida" de como salvar programas POKE 59466, Ø : POKE 59467, Ø : POKE 59468, 12 (RETURN)
para devolverle la posibilidad de trabajar correctamente con el cassette.



La subrutina 1000 ha sido modificada para tocar la nota mientras se computa la próxima. El resto en la linea 5000 permite corregir la terminación de la última nota.

```
10  REM EL COMMORODION
20  REM DEL TEMA 'SCOTT JOPLING'
30  REM ARTICULO DE ROM PAG. 61 OCT. 1977
40  REM CODIFICADO POR GREGORY YOB
50  GOSUB 2000
60  REM LEER TIEMPO, CONVERTIRLO EN JIFFIES
70  READ TM
80  BT = 3600/ TM
90  REM FUNCION DE INTERVALO
100 DEF FNI(X) = INT (BT/X-5)
110 PRINT "PULSE UNA TECLA"
120 GETA$: IFA$ = "" THEN 120
130 REM DE COMPONER NOTA
140 READ S$
150 S$=S$ + " sp sp sp sp sp "  <-  sp = SPACE
160 A$= MID$ (S$,1,1)
170 B$= MID$ (S$,2,1)
180 C$= MID$ (S$,3,1)
190 D$= MID$ (S$,4,1)
200 E$= MID$ (S$,5,1)
210 REM VERIFICAR SE ACABO
220 IF A$= "X" THEN CLR: GOTO 50
230 REM CALCULAR NOTA
240 IF A$= "R" THEN N= Ø: GOTO 370
250 FOR J= 1 TO 12
260 IF A$= MID$ ("AABCCDDEFFGG",J,1) THEN 280
270 NEXT J: GOTO 140
280 N=J
290 REM SOSTENIDO OBEMOL
300 IF B$= "#"  THEN  N=N+1
310 IF B$="!"   THEN  N=N-1
320 REM CALCULO DE OCTAVA
330 Q= ASC(C$) - 48 AND 7
340 N=N -15+12*Q
350 IF N<1 OR N<36 THEN 140
360 REM DURACION
370 FOR J= 1 TO 5
380 IF  D$=MID$ ("WHQES"[redacted]HEN 400
390 NEXT J: GOTO 140
400 D= FNI(2↑*(J-1))
410 IF E$ =". "THEN D= 1.5*D
420 REM TOCAR NOTA
430 GOSUB 1000: GOTO 140
```

```
5000  REM FINAL
5010  DATA "R 1 W" , "X"


1000  REM TOCAR NOTA N DURANTE D JIFFIES
1010  REM ----- T1=TI (ATENCION !)
1020  POKE AF, P(N)
1030  POKE AW, W(N)
1050  IF TI - T1 < D THEN 1050
1060  T1=TI
1070  RETURN
1100  REM DATOS DE FRECUENCIA
1110  DATA  227,211, 199, 187, 177,167
1120  DATA  157,148, 140, 132, 124,117
2000  REM INICIALIZACION
2010  DIM P (36), W(36)
2020  W= 15
2030  FOR J=1 TO 36
2040  IF J= 13 THEN RESTORE : W=51
2050  IF J= 25 THEN RESTORE : W=85
2060  READ P (J) : W(J) = W:NEXT J
2070  AW = 59466: AF: 59464
2080  POKE 59467, 16
2090  RETURN
3000  REM DATOS DE LA CANCION
3010  REM TIEMPO DE LOS GOLPES POR MINUTO
3020  DATA 80
3030  REM NOTAS A LA MUSICA
3050  DATA D 2S, E 2S, C 2S, A 2E, B 2S, A 2S
3040  DATA D 3S, E 3S, C 3S, A 3E, B 3S, G 3E
3060  DATA  A 2S, G 1Q, G 3E, D 1S, D# 1S
3070  DATA  E 1S, C 2E, E 1S, C 2E, E 1S, C 2Q
3080  DATA  C 3S, D 3S, D:: 3S
3090  DATA  E 3S, C 3S, D 3S, E 3E
3100  DATA  B 3S, D 3E, D 3Q, D 1S, D # 1S
3110  DATA  E 1S, C 2E, E 1S, C 2Q
3120  DATA  C 2S, A 3S, G 2S
3130  DATA  F #2S, A 3S, C 3S,E 3E, D 3S, C 3S
3140  DATA  A 3S, D 3Q, D 1S, D:: 1S
3150  DATA  E 1S, C 2E, E 1S, C 2E, E 1S, C 2Q,
3160  DATA  C 3S, D 3S, D #3S
3170  DATA  E 3S, C 3S, D 3E, E 3E, B 3S, D 3E
3180  DATA  C 3Q,,C 3S, D 3S, (note the ","!)
3190  DATA  E 3S, C 3S, D 3S, E 3E, C 3S, D 3S
3200  DATA  C 3S, E 3S, C 3S, D 3S, E 3E, C 3S
3210  DATA  D 3S, C 3S, E 3S, C 3S, D 3S, E 3E
3220  DATA  B 3S, D 3E, C 3Q, C 3S, E 2S, F 2S
3230  DATA  F #2S, G 2S, A 3S, G 2E, E 2S, F 2S
3240  DATA  F #2S, G 2E, A 3S, G 2E, E 2S, C 2S
3250  DATA  B 1S, A 2S, B 2S, C 2S, D 2S, E 2S
3260  DATA  D 2S, C 2S, D 2S
3270  DATA  C 2E, G 1E, C 1E
```

ALTA RESOLUCION
Por John R. Sherburne
Febrero, 1979

```
033A                          ORG      $033A

033A    A9 00    START        LDAIM    $00       INITIALIZE
033C    85 53                 STA      $0053
033E    85 56                 STA      $0056
0340    38                    SEC
0341    A5 51                 LDA      $0051
0343    E9 4f                 SBCIM    $4F       CHECK VALID X
0345    30 03                 BMI      CHECK
0347    E6 54                 INC      $0054
0349    60                    RTS

034A    38       CHECK        SEC                CHECK VALID Y
034B    A5 52                 LDA      $0052
034D    E9 31                 SBCIM    $31
034F    30 03                 BMI      HALF
0351    E6 55                 INC      $0055
0353    60                    RTS

0354    46 51    HALF         LSR      $0051
0356    90 02                 BCC      NOCAR
0358    E6 56                 INC      $0056
035A    46 52    NOCAR        LSR      $0052
035C    90 04                 BCC      NOCRY
035E    E6 56                 INC      $0056
0360    E6 56                 INC      $0056     DIVIDE X & Y /2
0362    A9 01    NOCRY        LDAIM    $01       DET. QUAD N: POINT
0364    A4 56    LOOP         LDY      $0056     QUAD Nº IN $0056
0366    F0 06                 BEQ      MATCH
0368    0A                    ASLA
0369    C6 56                 DEC      $0056
```

...//...

.../...

```
036B   4C 64 03             JMP     LOOP
036E   85 56      MATCH     STA     $0056
0370   06 52                ASL     $0052
0372   06 52                ASL     $0052
0374   06 52                ASL     $0052
0376   A5 52                LDA     $0052
0378   06 52                ASL     $0052
037A   26 53                ROL     $0053     Y x dec. 40.
037C   06 52                ASL     $0052     (Nº CHAR/LINE)
037E   26 53                ROL     $0053
0380   62 52                ADC     $0052
0382   85 52                STA     $0052
0384   A5 53                LDA     $0053
0386   69 00                ADCIM   $00
0388   85 53                STA     $0053
038A   A5 52                LDA     $0052
038C   65 51                ADC     $0051     ADD X & Y x 40
038E   85 52                STA     $0052
0390   90 02                BCC     NOCHG
0392   E6 53                INC     $0053
0394   18         NOCHG     CLC
0395   A9 80                LDAIM   $80
0397   65 53                ADC     $0053
0399   85 53                STA     $0053
039B   A0 10                LDYIM   $10
039D   A2 00                LDXIM   $00
039F   A1 52                LDAIX   $0052
03A1   88         CHARAC    DEY
03A2   D9 BA 03             CMPY    TABLE
03A5   F0 09                BEQ     FOUND
03A7   C0 00                CPYIM   $00
03A9   D0 F6                BNE     CHARAC
03AB   A6 B1                LDX     $00B1     CHECK FOR $B1 FOR
03AD   F0 01                BEQ     FOUND     USR ARGUMENT
03AF   60                   RTS

03B0   98         FOUND     TYA
03B1   05 56                ORA     $0056
03B3   A8                   TAY
03B4   B9 BA 03             LDAY    TABLE     TABLE OF ALL 16
03B7   81 52                STAIX   $0052     CHARACTERS, one
03B9   60                   RTS               to then screen.

03BA   20         TABLE     =       $20
03BB   7E                   =       $7E
03BC   7C                   =       $7C
03BD   E2                   =       $E2
03BE   7B                   =       $7B
03BF   61                   =       $61
03C0   FF                   =       $FF
03C1   EC                   =       $EC
03C2   6C                   =       $6C
```

.../...

...//...

03C3	7F			
03C4	E1		=	$7F
03C5	FB		=	$E1
03C6	62		=	$FB
03C7	FC		=	$62
03C8	FE		=	$FC
03C9	A0		=	$FE
	0000	.END	=	$A0

UTILIZACION DE LA SUBRUTINA DE ALTA RESOLUCION

En primer lugar, es necesario cargarla en el PET, en el espacio de memoria correspondiente al buffer del 2º cassette, tal como se aprecia en el listado de assembler. Para ello y como reto a su paciencia, puede optar por transformar los códigos hexadecimales en decimales y, a continuación, POKARLO en la memoria (terrible!!!)... También se puede utilizar el programa en BASIC "CARGADOR DE HEXA", existente en la librería de programas.
Otro sistema oportuno es el de introducirlo mediante el MONITOR y despues salvarlo en cassette en hexa, a traves del mismo MONITOR.
Para los que no deseen tomarse esas molestias, disponemos de cassettes con la subrutina grabada, al nuevo precio estandard de 500 pts. (cassette no incluido); el nombre en la librería es SUBRUTINA DE GRAFICAS-3.

Lapantalla queda dividida en 4000 puntos, contra los 1000 habituales del PET.
Asi los ejes quedan establecidos en 80 x 50

Encendido de un punto en la pantalla:
POKE 81,X:POKE 82,Y:A=USR(0)

EJEMPLO CON LAS FIGURAS DE LISSAJOUS

```
10 POKE 1,58:POKE 2,3:PRINT"  "        (borrado pantalla)
20 DELTA=2    /900
30 P=3:Q=4
40 FOR I=0 TO 900
50 THETA=DELTA*I
60 X=INT(39.5+38*COS(P*THETA))
70 Y=INT(25.5+24*SIN(Q*THETA))
80 POKE 81,X:POKE 82,Y:A=USR(o)
90 NEXT I
100 GET A$:IF A$="" THEN 100
```

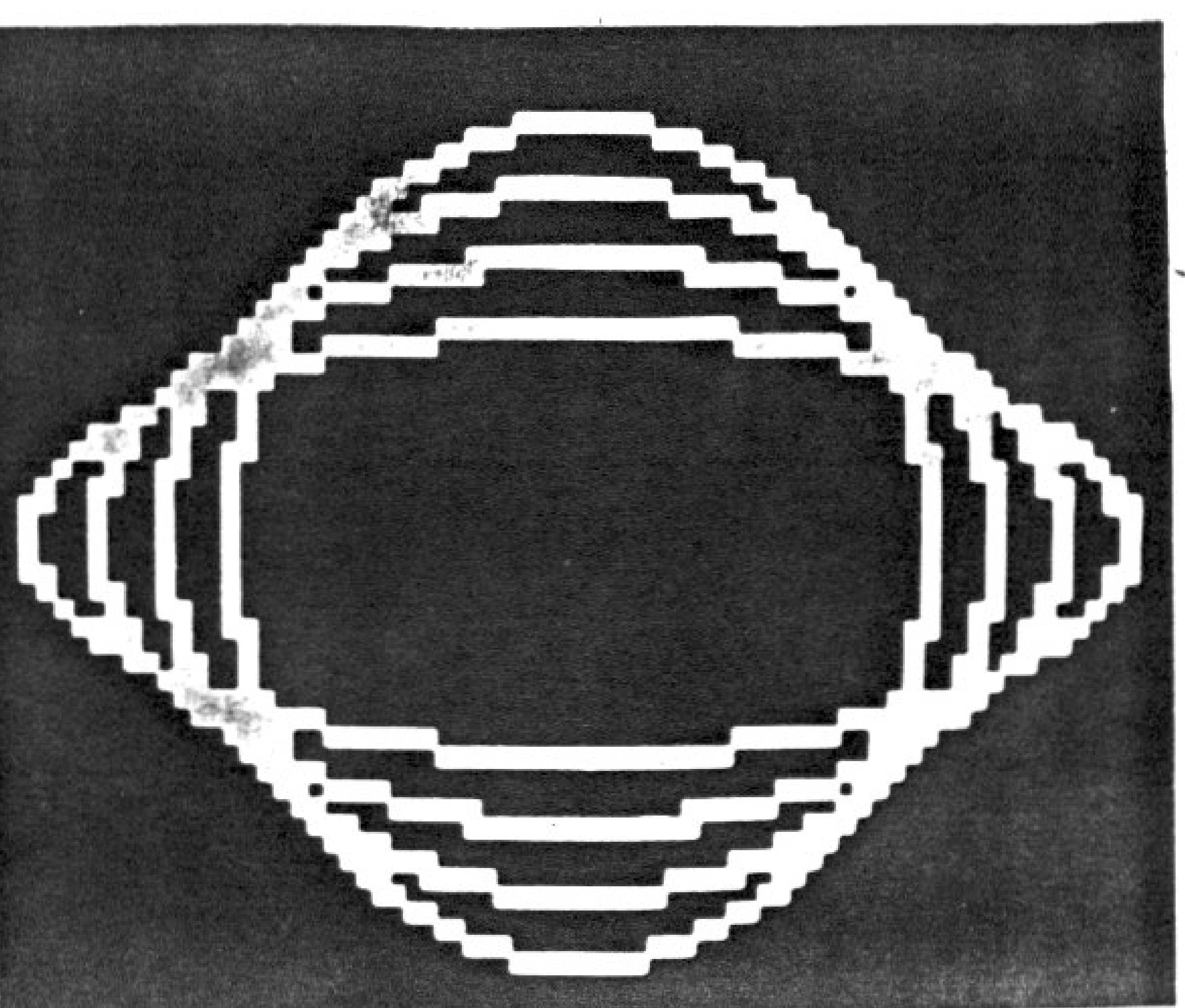
En las 2 páginas siguientes van ó lindos ejemplos de como hacer figuras en el PET que hasta ahora hubiesen parecido imposibles...

RUN smart service!!!

NOTA:
"clr" significa:
Borrado pantalla

```
1    POKE 1,58:POKE 2,3
10   PRINT "(clr)"
20   FOR R=4 TO 16 STEP 4
30   P=38-R
40   Q=8+R
50   F=2*π/300
60   FOR I=0 TO 300
70   AN=F*I
80   X=INT(39.5+P*COS(AN))
90   Y=INT(24.5+Q*SIN(AN))
100  POKE 81,X:POKE 82,49-Y:A=USR(0)
110  NEXT I
120  NEXT R
130  GET G$:IF G$="" GOTO 130
```

Figura 1



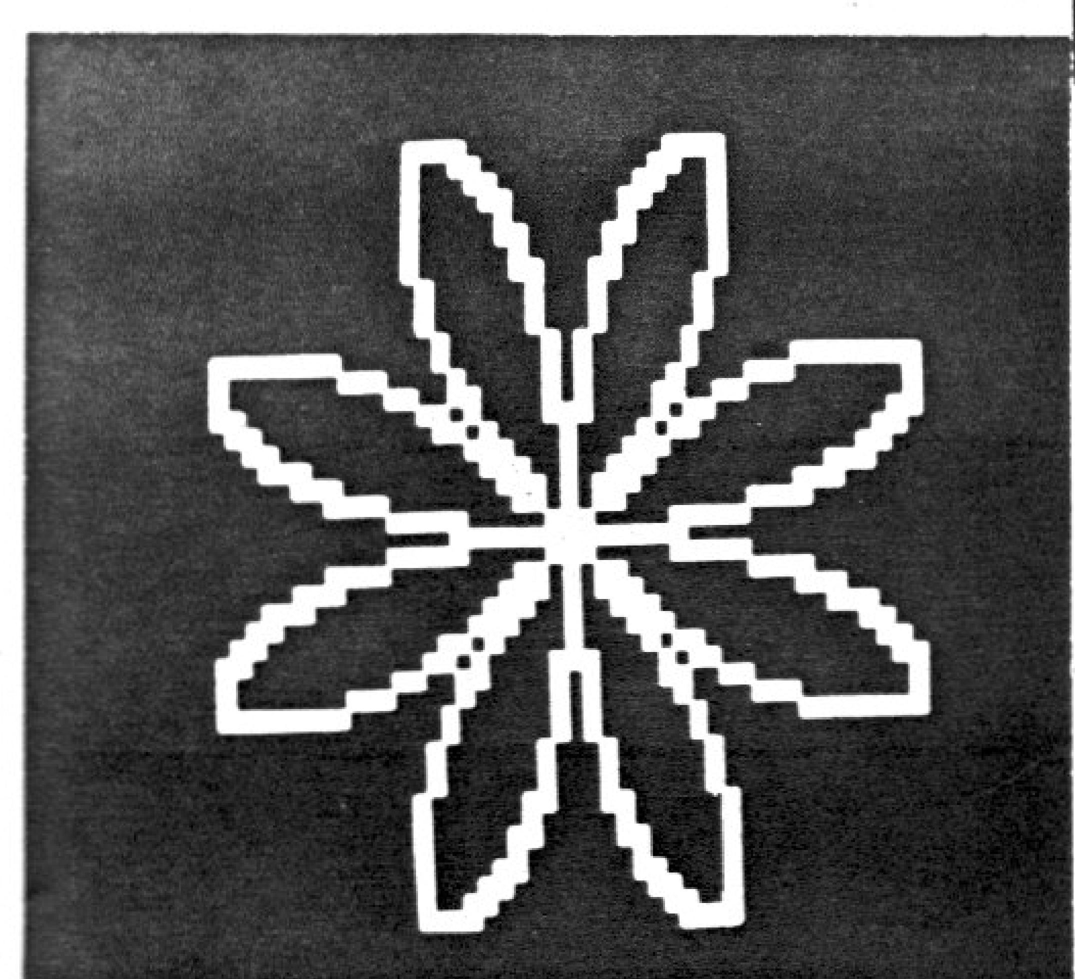


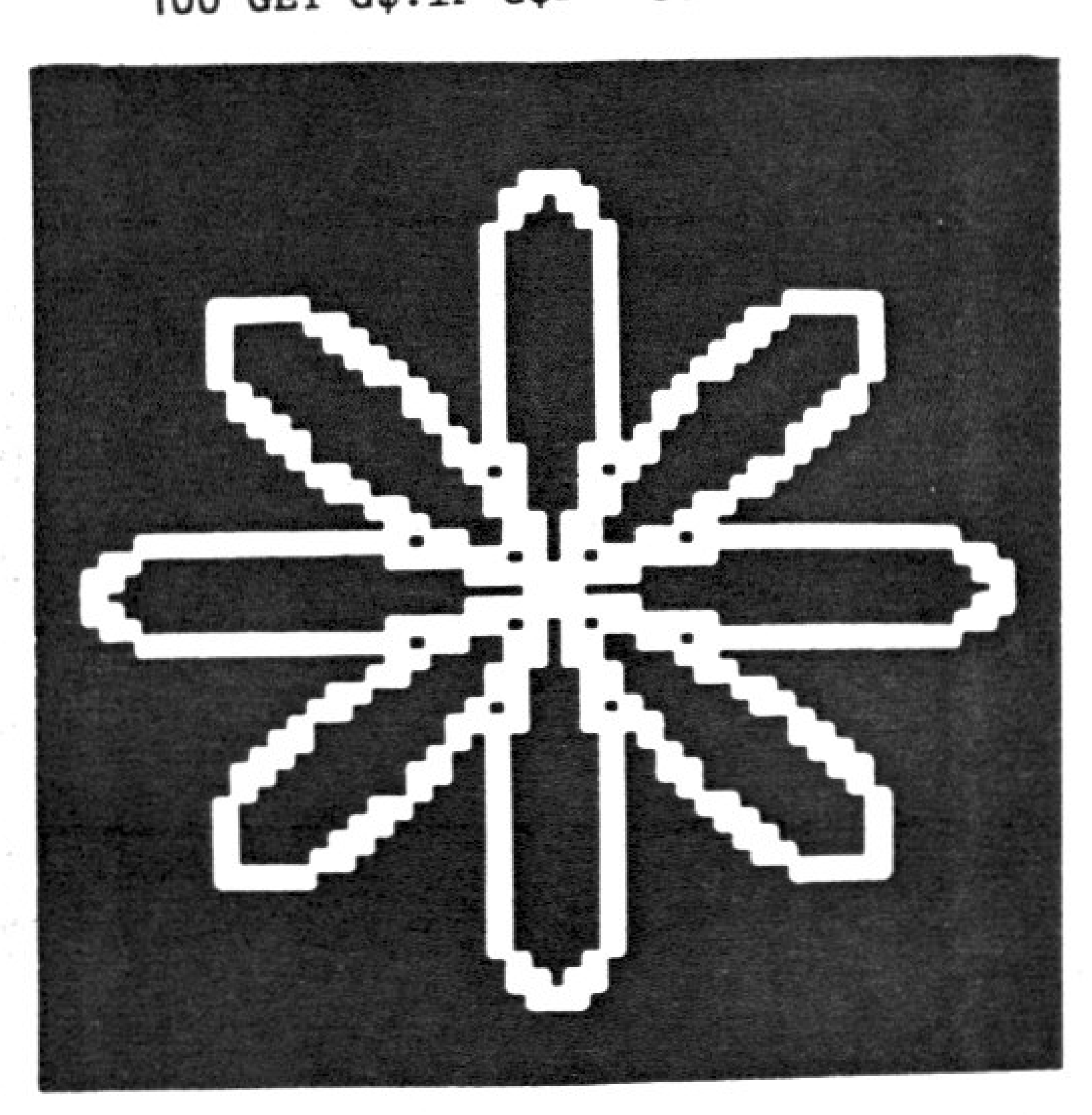
Figura 2A

```
1    POKE 1,58:POKE 2,3
10   PRINT "(clr)"
20   P=24:N=4
30   F=2*π/600
40   FOR I=0 TO 600
50   AN=I*F
55   R=P*SIN(N*AN)
60   X=INT(R*COS(AN)+39.5)
70   Y=INT(R*SIN(AN)+24.5)
80   POKE 81,X:POKE 82,49-Y:A=USR(0)
90   NEXT I
100  GET G$:IF G$="" GOTO 100
```

Figura 2B

Cambiar respecto a Fig. 2A solamente:

```
55 R=P*COS(N*AN)
```



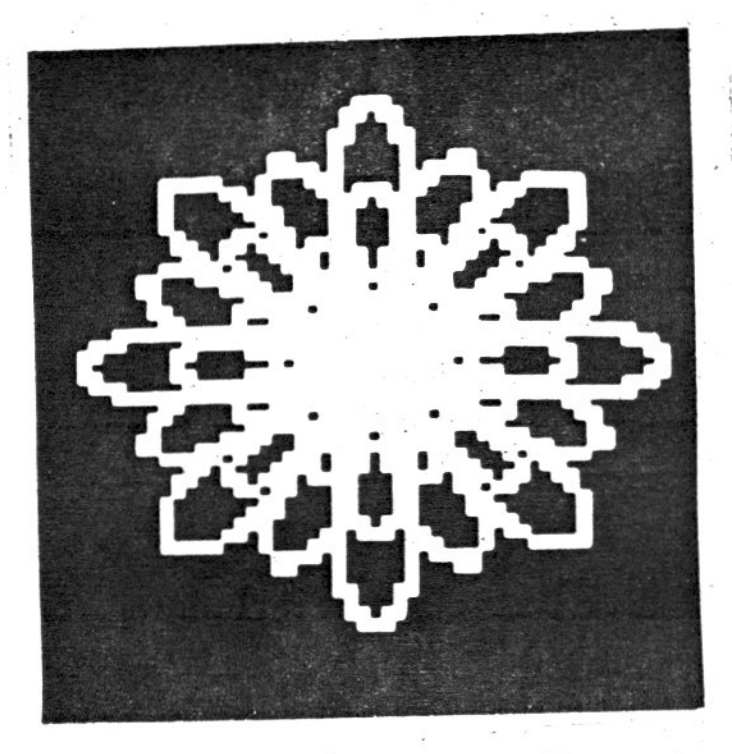


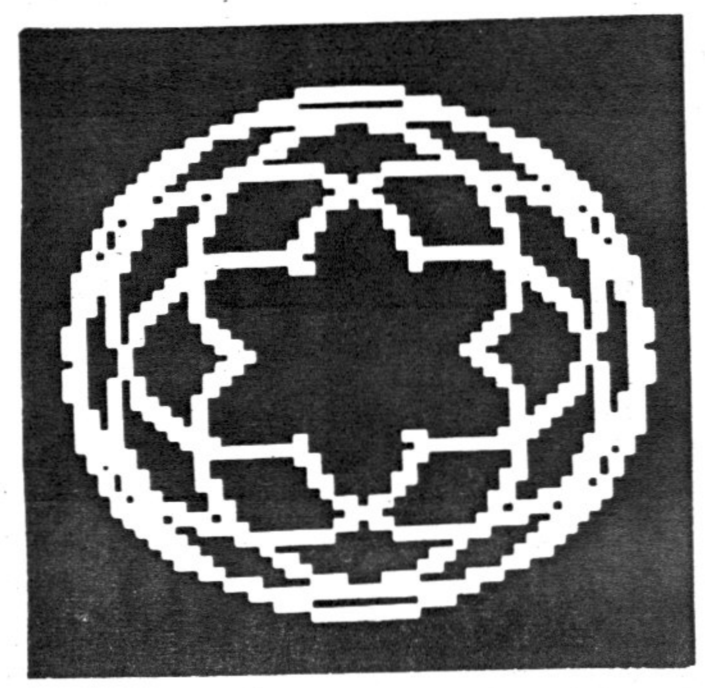
Figura 2C

(Changes to 2B only)

```
20  N=4
31  K1=1
32  FOR K2=0 TO 20 STEP 4
33  P=24-K2
34  K1=K1*-1
55  R=P*SIN(N*AN)
56  IF K1<0 THEN R=P*COS(N*AN)
```

Figura 3

```
1   POKE 1,58:POKE 2,3
10  PRINT "(clr)"
20  P=9;Q=15/2
30  F=2*π/250
40  FOR I=0 TO 1250
50  AN=I*F
60  X=(P+Q)*COS(AN)+Q*COS((P+Q)*AN/Q)
70  Y=(P+Q)*SIN(AN)+Q*SIN((P+Q)*AN/Q)
80  X=INT(X+39.5):Y=INT(Y+24.5)
90  POKE 81,X:POKE 82,Y:A=USR(0)
100 NEXT I
110 GET G$: IF G$="" GOTO 110
```



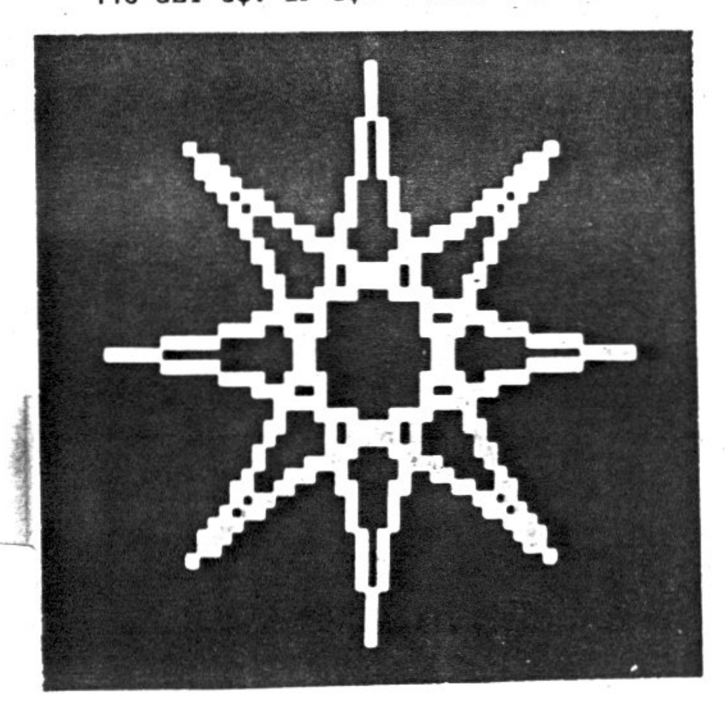
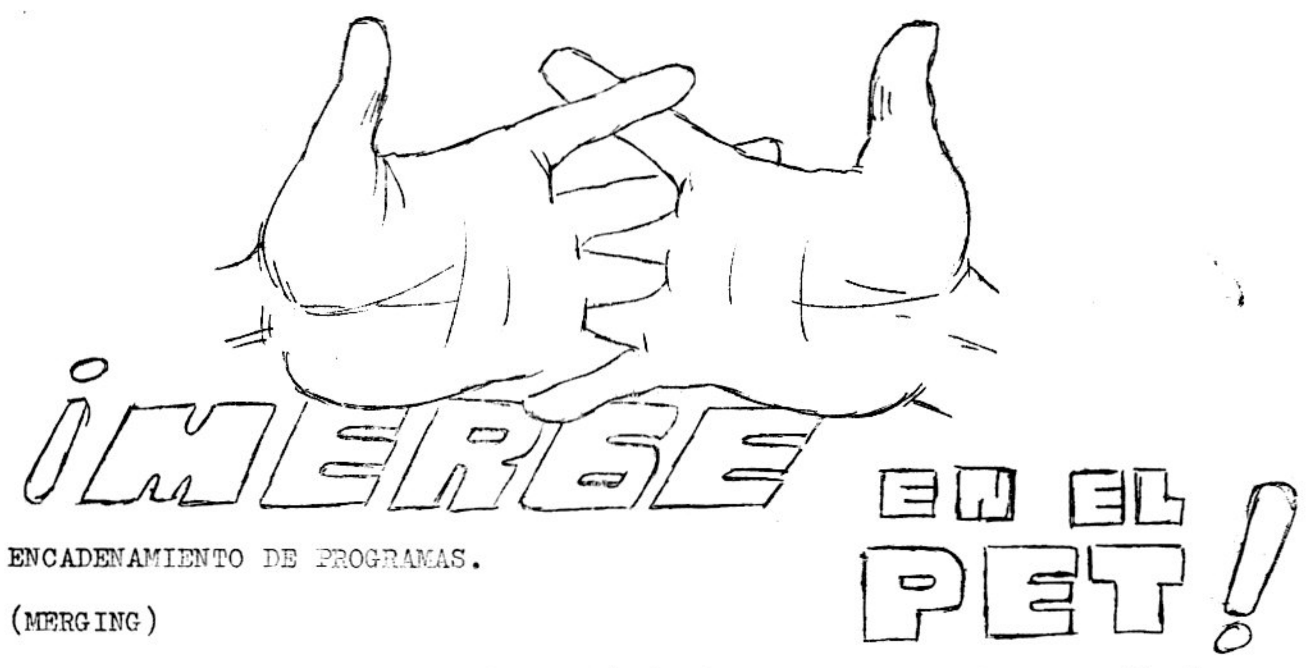


Figura 4

```
1   POKE 1,58:POKE 2,3
10  PRINT "(clr)"
20  P=24:Q=9
22  DT=300
24  F=2*π/DT
28  FOR I=1 TO 25
30  DL=P*I/Q-INT(P*I/Q)
32  IF DL<.00001 GOTO 36
34  NEXT I
36  PT=I*P/Q
38  PRINT "(home)";INT(PT+.5)
40  FOR J=0 TO I*DT
50  AN=J*F
60  X=(P-Q)*COS(AN)+Q*COS((P-Q)*AN/Q)
70  Y=(P-Q)*SIN(AN-Q*SIN((P-Q)*AN/Q)
80  X=INT(X+39.5):Y=INT(Y+24.5)
90  POKE 81,X:POKE 82,Y:A=USR(0)
100 NEXT J
110 GET G$:IF G$="" GOTO 110
```



ENCADENAMIENTO DE PROGRAMAS.

(MERGING)

Los ppogramas profesionales están hechos frecuentemente a partir de una librería de subrutinas standard las cuales, se encadenan convenientemente para formar el programa definitivo.

Hasta el presente esta técnica era inalcanzable con el PET. No obstante, describiremos a continuación un método para juntar la memoria del PET dos o más programas provenientes de sendos cassettes (siempre, claro está, que los números de líneas no sean coincidentes).

Con este procedimiento nos evitamos la engorrosa tarea de tener que entrar un programa normalmente, por cassette y, a continuación entrar el siguiente a mano por el teclado.

Lo primero que debemos hacer es prepararlos programas en el momento de grabarlos en el cassette de modo que, al ser leidos posteriormente no machaquen al programa existente en la memoria del PET.

Ello lo condiguiremos de la siguiente manera:

- CARGAR EL PROGRAMA A PREPARAR EN EL PET DE LA FORMA HABITUAL (LOAD, etc.)
- COLOCAR UNA CINTA VIRGEN EN EL CASSETTE.
- OPEN 1,1,1: CMD 1: LIST (RETURN).
- PET solicitará como habitualmente que se pulsen las teclas de grabación y el PET retorna a su modo de operar normal.
- PRINT ## 1 CLOSE 1 (RETURN)

A partir de ese momento ya disponemos de un cassette listo para añadir su contenido a cualquier programa existente en el PET; ello lo haremos de la siguiente manera:

- Tener el PET cargado con un programa.
- Inserte la cinta preparada en el cassette.
- OPEN 1 (RETURN)
- Pulse "Play" tal como lo indica el PET.

Y ahora siga exactamente los pasos que se detallan:

- Borre la pantalla.
- Baje el cursor 4 veces.

Escriba la siguiente línea SIN PULSAR RETURN.

- POKE 611,1: POKE 525,1: POKE 527,13: ?"S" (S, significa, cursor a casa).
- No pulse RETURN.
- Pulse cursor a casa.
- Pulse 6 bajadas de cursor.
- Reproduzca de nuevo la línea anterior.
- Pulse RETURN y vea como el cassette se pone en marcha.

Cuando el programa ha entrado por completo aparecerá un mensaje de ? SINTAX ERROR O ?OUT OF DATA ERROR entre las dos líneas. Si no apareciera dicho mensaje pulse RUN-STOP.

El PET retornará a su modo normal de operación haciendo:

CLOSE 1 (RETURN)

A partir de ese momento el contenido del último cassette habrá sido añadido al programa existente en el PET siempre, por supuesto, que no hubiera líneas coincidentes.

Es una buena práctica el prepararse mediante este procedimiento una librería de subrutinas standard confeccionados con números de líneas diferentes.

Se debe tener en cuenta también que el uso de las mismas variables en las diferentes subrutinas standard(confeccionadas con números de líneas DIFERENTES) puede dar lugar a confusión en el programa general si no se tiene en cuenta a la hora de confeccionarlos.

DISTRIBUCION DE LA MEMORIA EN EL PET

BLOQUE	TIPO	DIRECCION INICIAL	FUNCION
0	RAM	$0000	Zona de trabajo/almacen texto BASIC
1	RAM	$1000	Almacen texto BASIC (solamente para 8K)
2	--	$2000	Expansion RAM
3	--	$3000	Expansión RAM
4	--	$4000	Expansión RAM
5	--	$5000	Expansión RAM
6	--	$6000	Expansión RAM
7	--	$7000	Expansión RAM
8	RAM	$8000	Memoria de pantalla (1k)
9	--	$9000	Expansión ROM
10	--	$A000	Expansión ROM
11	--	$B000	Expansión ROM
12	ROM	$C000	BASIC (principalmente interpretador órdenes)
13	ROM	$D000	BASIC (principalmente operaciones mátemáticas)
14	ROM	$E000	Editor de pantalla
	I/O	$E800	Todas las entradas y salidas internas del PET
15	ROM	$F000	Diagnosticos del sistema operativo

BLOQUE Ø DE PAGINAS DE 256 BYTES

PAGINA	TIPO	DIRECCION INICIAL	FUNCION
0	RAM	0000	Zona de almacenamiento del sistema operativo del BASIC
1	RAM	0100	Stack
2	RAM	0200	Almacenamiento de trabajo del sistema operativo
3	RAM	0300	Buffers de Cassette
15	RAM	0400	Area texto del BASIC

Ver descripción detallada por páginas

BLOQUE 14 EN SEGMENTOS DE 2K

PAGINA	TIPO	DIRECCION INICIAL	FUNCION
0	ROM	$E000	Editor de pantalla
1	I/O	$E800	PET I/O
0	PIA	$E810	Teclado
1	PIA	$E820	IEEE-488
2	VIA	$E840	USR PORT, cassettes

Nora.- Todas las direcciones están expresadas notación hexadecimal.($)

DESCRIPCION DETALLADA DE LAS POSICIONES DE MEMORIA DE LA PAGINA Ø DEL PET:

(Las direcciones no especificadas se utilizan tambien pero no tienen una aplicación concreta)

DESDE	HASTA	FUNCION
ØØØ	---	$4C (instrucción de JMP para la función USR)
ØØ1	ØØ2	Dirección de salto para la función USR
		Area de control de entrada/salida:
ØØ3	---	Número del canal de entrad/salida activado
ØØ4	---	Impresión de espacios blancos para CRLF (no se utiliza)
ØØ5	---	Próxima columna de Basic a imprimir
ØØ6	---	Tamaño del terminal (no se utiliza)
ØØ7	---	Limite en el barrido de columnas (sin utilizar)
ØØ8	---	Número de la línea almacenada anteriormente en el buffer
ØØ9	---	$2C (ASCII de la coma para el proceso de INPUT de más de una variable)
Ø1Ø	Ø89	Buffer de entrada del BASIC (8Ø bytes)
Ø9Ø	---	Contador general del BASIC
Ø91	---	$ØØ que se utiliza como delimitador
Ø92	---	Contador utilizado por el BASIC
		Area de evaluación de variables:
Ø93	---	Flag que indica que hay variables dimensionadas
Ø94	---	Flag de clase de variable (Ø = númerica,1 = cadena)
Ø95	---	Flag de variables enteras(%)
Ø96	---	Flag de protección de palabras reservadas (comandos del BASIC)
Ø97	---	Flag de variables suscritas (permite el parentesis sin Err.de Sintaxis)
Ø98	---	Flags de input o READ
Ø99	---	Flag de signo en TAN
1ØØ	---	Flag de supresión de salidas (+ normal, - suprimida)
1Ø1	---	Indice del siguiente descriptor disponible
1Ø2	1Ø3	Puntero del último $ (variable de cadena)
1Ø4	111	Tabla de descriptores que apuntan a las variables
112	113	Indice indirecto número 1
114	115	" " " 2
116	121	Pseudo registro para los operandos de las funciones
		Area de control de variables:
122	123	Puntero que señala el comienzo del texto de BASIC
124	125	" " " " " de la zona de variables
126	127	" " " " " de las tablas de variables dimensionadas
128	129	Puntero que señala el final de la zona de variables
13Ø	131	" " " " comienzo de la zona de variables de cadena
132	133	" " " " espacio final de una variable de cadena
134	135	" " " la posición más alta de RAM disponible
136	137	Número de la línea que está siendo ejecutada (un Ø significa comando directo)
138	139	Número de la línea que tiene que arrancar el comando CONT
14Ø	141	Puntero que señala a la siguiente instrucción a ejecutar
142	143	Número de la línea de DATA (usado en control de errores)
144	145	Puntero que señala el dato a leer en líneas DATA

Area de evaluación de expresiones:

DESDE	HASTA	FUNCION
146	147	Fuente de INPUT
148	149	Nombre de la variable con que se está trabajando en ese momento
150	151	Puntero que señala a la variable en la memoria
152	153	Puntero que indica la variable utilizada en el FOR-NEXT que esta ejecutando.
154	155	Puntero que indica el operador que se está utilizando
156	—	Mascara especial para el operador que se está utilizando
157	158	Puntero que señala la función DEF
159	160	" " " " descripción de una variable de cadena
161	—	Longitud de la variable anteriormente descrita
162	—	Constante utilizada en la recolección de datos auxiliares
163	—	$4C (Instrucción JMP utilizada por el BASIC)
164	165	Vector para la ejecución de funciones
166	171	Acumulador flotante número 3
172	173	Puntero de transferencia de bloques número 1
174	175	Puntero de transferencia de bloques número 2
176	181	Acumulador flotante número 1 (Aquí se evalúa USR)
182	—	Duplicado del signo de la mantisa del acum.fl.num.1
183	—	Contador de bits en desplazamientos de normalización del acumulador flotante número 1
184	189	Acumulador flotante número 2
190	—	Byte de rebase de capacidad en el argumento flotante
191	—	Duplicado del signo de la mantisa
192	193	Puntero del repertorio del ASCII para las rutinas de conversión

Area de subrutinas de RAM :

DESDE	HASTA	FUNCION
194	199	Código CHRGOT RAM, Recoge el siguiente caracter del texto de BASIC
200	—	Código CHRGOT RAM. " " caracter actual del texto de BASIC
201	202	Puntero que señala el texto de BASIC
203	223	Próximo número aleatorio preparado para salir (Función RND)

Area de almacenamiento del sistema operativo:

DESDE	HASTA	FUNCION
224	225	Puntero que señala la línea de arranque del cursor
226	—	Posición del cursor(columnas)
227	228	Dirección indirecta para usos generales
229	233	" directa " " "
234	—	Flag de entrecomillado(movimientos cursor, etc. programados en PRINT)
238	—	Longitud del nombre del fichero actual
239	—	Número de fichero actual
240	—	Dirección primaria actual
241	242	" secundaria actual.
243	244	Puntero que señala el inicio del buffer de cassette utilizado
245	—	Número actual de línea en pantalla
246	—	Almacen temporal de dato en entrada/salida
247	248	Puntero que señala al principio del programa del sistema operativo
249	250	Puntero que señala el nombre de fichero actual
251	254	Bytes sin utilizar (libres de trabajo)
255	—	Byte de exceso de capacidad usado en conversiones ASCII

PAGINA 1 DEL PET

Los primeros 62 bytes de la página 1 se utilizan para la correción de errores em la lectura de cassettes y registros de expansión en las conversiones ASCII.

El resto de la página 1 sirve para el almacenamiento de direcciones de retorno en las intrucciones del BASIC:GOSUB y FOR-NEXT.

También se utiliza como STACK de trabajo de la CPU.

DESCRIPCION DETALLADA DE LAS POSICIONES DE MEMORIA DE LA PAGINA 2 DEL PET:

DESDE	HASTA	FUNCION
512	514	Reloj de 24 horas en 1/60 de segundo (binario)
517	518	Factor de correción para el reloj
519	520	Flag de interrupción de los interruptores de los cassettes
523	--	" " VERIFY (impide la carga del cassette en memoria)
524	--	Palabra de status (ST)
525	--	Indice del buffer de teclado
526	--	Flag que indica que se esta en impresión en reverso
527	536	Buffer del teclado (las teclas pulsadas se almacen aquí)
537	538	Vector de solicitud de interrupciones en RAM
539	540	Instrucción BRK para el vector de interrupciones en RAM
549	--	Contador para el temporizador de parpadeo del cursor
550	--	Byte sin utilizar (libre de trabajo)
551	--	Flag de parpadeo del cursor
553	577	Tabla de direcciones de las líneas de la pantalla(25)
578	587	Tabla de direcciones lógicas
588	597	Tabla de direcciones primarias
598	609	Tabla de direcciones secundarias
610	--	Indice de las tablas anteriores
611	--	Defecto del aparato de entrada número ......
612	--	Defecto del aparato de salida número ....
613	--	Computo de paridad en escritura de cassettes
631	--	Contador de bloques redundantes en cassette
624	--	Contador para temporización en escritura de cassette
625	626	Indice del siguiente caracter en los buffers de cassette
627	--	Contador de temporización en lectura de cassette
628	--	Flag de indicación de error de cinta
629	--	" " " " rutina de lectura de nombres abreviados (ficheros)
630	631	Indice de las direcciones a corregir en la lectura de cinta
632	--	Flag de lectura de cassette
633	--	Contador de segundos en escritura de cassettes (espacios en blanco)
634	825	Buffer del cassette número 1 (192 bytes)
826	1017	" " " " 2 " "
1018	1023	No se utilizan

Mediante este programa se renumeraran las lineas de un programa ya existente en la memoria del PET.
Asignará a la primera linea el numero 10 he irá incrementando de 10 en 10.
El proceso para renumerar un programa existente:
1º) Cargar el programa a renumerar.
2º) Cargar el programa renumerador sin borrar el anterior, haciendo un MERGE (se explica en otras páginas del boletín)
3º) RUN 63000.
Esperar el tiempo necesario y nuestro programa quedará renumerado.

LISTADO BASIC

```
63000  REM PARA RENUMERAR "RUN 63000"
63010  DIM OL(255): OM= 0: Al=1: Ah=4: NL=10: NH=0
63020  AD= 256.AH + AL: LL=PEEK (AD-2): LH=PEEK(AD-3): OL=256. LH-LL
63030  IF OL=63000 GOTO 63500:
63040  OL (OM) = OL: OM=OM-1
63050  POKE AD-2, NL: POKE AD -3, NH: NL=NL-10: IF NL>255  THEN
       NL=NL-256: NH=NH-1
63060  IF  OM   255 GOTO 63500:
63070  AL=PEEK  (AD): AH=PEEK (AD-1): GOTO 63020
63500  REM
63510  PRINT OM;  "NUMEROS DE LINEA CORREGIDOS": OM=OM-1: L=1024
63520  L=L+4: LN= 256. PEEK (L)PEEK(L-1): IF LN=63000THEN PRINT
       "FINAL" END
63530  L=L-1: CH=PEEK(L) : IF CH = 0 GOTO  63520
63540  IF (CH <  137) AND(CH    141) AND(CH -  167)  GOTO 63530
63550  LO=L
63560  L=L-1: CH=PEEK(L): IF CH=32 GOTO 63560
63570  IF (CH  47)  AND (CH  58) THEN GOSUB 63700: GOTO 63560
63580  IF N$=""  GOTO 63530
63590  IF  CH=44 THEN GOSUB 63800: GOTO 63550
63600  GOSUB 63800: GOTO 63530
63610  GOTO  63530
63700  N=CH-48: N$=N$+RIGHT$  (STR$(N), 1): RETURN
63800  J=-1: N= VAL (N$):  FOR I=0 TO OM: IF OL(I) =N THEN J=I: I=OM
63810  NEXT I: IF J=-1 THEN PRINT "NO HALLADA LINEA :: ;N; EN TABLA"
       GOTO 63890
63820  NL= 10 . J-10: NL$=STR$ (NL): NL$= RIGHT$ (NL$, LEN (NL$)-1)
63830  IF LEN (NL$) > (L-LO-1) THEN PRINT "NO HAY ESPACIO PARA
       CAMBIAR"; N; "POR"  NL: GOTO 63890
63840  IF LEN (NL$) > (L-LO-1) THEN NL$=NL$+" " : GOTO 63840
63850  FOR I=LO -1 TO L-1; N$=MID$ (NL$, I-LO, 1)· N=VAL(N$) +48:POKE I,N
63860  IF N$=" "THEN POKE I, 32
63870  NEXT I
63890  N$=" ":  RETURN
```

TABLA DE EQUIVALENCIAS ENTRE EL CODIGO HEXADECIMAL

E INSTRUCCIONES DEL PET

HEX	COMANDO	HEX	COMANDO
8Ø	END	A6	SPC
81	FOR	A7	THEN
82	NEXT	A8	NOT
83	DATA	A9	STEP
84	INPUT #	AA	+
85	INPUT	AB	-
86	DIM	AC	*
87	READ	AD	/
88	LET	AE	↑
89	GOTO	AF	AND
8A	RUN	BØ	OR
8B	IF	B1	>
8C	RESTORE	B2	=
8D	GOSUB	B3	<
8E	RETURN	B4	SGN
8F	REM	B5	INT
9Ø	STOP	B6	ABS
91	ON	B7	USR
92	WAIT	B8	FRE
93	LOAD	B9	POS
94	SAVE	BA	SQR
95	VERIFY	BB	RND
96	DEF	BC	LOG
97	POKE	BD	EXP
98	PRINT #	BE	COS
99	PRINT	BF	SIN
9A	CONT	CØ	TAN
9B	LIST	C1	ATN
9C	CLR	C2	PEEK
9D	CMD	C3	LEN
9E	SYS	C4	STR$
9F	OPEN	C5	VAL
AØ	CLOSE	C6	ASC
A1	GET	C7	CHR$
A2	NEW	C8	LEFT$
A3	TAB	C9	RIGHT$
A4	TO	CA	MID$
A5	FN	FF	(PI)

= = = = = = = =

LOS RESTANTES CODIGOS (DESDE CB HASTA FE, NO SON UTILIZADOS PARA FUNCIONES DEL PET ACTUALMENTE.

LINE	LOC	CODE	LINE
ØØØ2	ØØØØ		COPYRIGHT 1978 BY
ØØØ3	ØØØØ		COMMODORE INTERNATIONAL LIMITED
ØØØ5	ØØØØ		VARTAB = $7C
ØØØ6	ØØØØ		TXTPT =$CA
ØØØ7	ØØØØ		NCMDS =8
ØØØ8	ØØØØ		UPLI = $91
ØØØ9	ØØØØ		RDT = $FFCF
ØØ1Ø	ØØØØ		URT = $FFD2
ØØ11	ØØØØ		CBINV = $021B
ØØ12	ØØØØ		UARM = $C38B
ØØ13	ØØØØ		FA = $F1
ØØ14	ØØØØ		FNLEN = $EE
ØØ15	ØØØØ		FNADR = $F9
ØØ16	ØØØØ		STAL = $F7
ØØ17	ØØØØ		STAH = $F8
ØØ18	ØØØØ		EAL = $E5
ØØ19	ØØØØ		EAH = $E6
ØØ2Ø	ØØØØ		ZERO PAGE MONITOR RESERVE AREA
ØØ21	ØØØØ		*=$ØA
ØØ22	ØØØØ		
ØØ23	ØØØA		WRAP *=*+1
ØØ24	ØØØB		DIFF *=*+2
ØØ25	ØØØD		BRKF *=*+1
ØØ26	ØØØE		PREYC *=*+2
ØØ27	ØØØF		ACMD *=*+2
ØØ28	ØØ11		TMPO *=*+2
ØØ29	ØØ13		TMP2 *=*+2
ØØ3Ø	ØØ15		TMP4 *=*+2
ØØ31	ØØ17		TMP6 *=*+2
ØØ32	ØØ19		PCL *=*+1
ØØ33	ØØ1A		PCH *=*+1
ØØ34	ØØ1B		FLGS *=*+1
ØØ35	ØØ1C		ACC *=*+1
ØØ36	ØØ1D		XR *=*+1
ØØ37	ØØ1E		YR *=*+1
ØØ38	ØØ1F		SP *=*+1
ØØ39	ØØ2Ø		SAVX *=*+1
ØØ4Ø	ØØ21		TMPC *=*+1
ØØ41	ØØ22		TMPC2 *=*+1
ØØ42	ØØ23		RCNT = TMPC
ØØ43	ØØ23		LCNT = TMPC2
ØØ44	ØØ23		ISTR *=*+16
ØØ45	ØØ33		*= $400
ØØ46	Ø4ØØ		ENTER COMPLIED BASIC TEXT
ØØ47	Ø4ØØ	ØØ	BYT Ø, 13, 4, 1Ø, Ø, 158
ØØ47	Ø4Ø1	ØD	

```
ØØ47      Ø4Ø2     Ø4
ØØ47      Ø4Ø3     ØA
ØØ47      Ø4Ø4     ØØ
ØØ47      Ø4Ø5     9E
ØØ48      Ø4Ø6     2Ø  31
ØØ48      Ø4ØC     ØØ
ØØ48      Ø4ØD     ØØ
ØØ48      Ø4ØE     ØØ
```

PET MONITOR 13. 1......... PAGE ØØØ2

```
LINE      LOC            CODE          LINE

ØØ49      Ø4ØF
ØØ5Ø      Ø4ØF                         CALL ENTRY POINT
ØØ51      Ø4ØF                              STACK CONTAINS y,X;A;S,
                                            PC
ØØ52      Ø4ØF                         BREAK ENTRY POINT
ØØ53      Ø4ØF                              STACK CONTAINS Y,X,A,
                                            S, PC
ØØ54      Ø4ØF
ØØ55      Ø4ØF     A9  27              CALLE     LDA #( BRKE
ØØ58      Ø411     8D  1B  Ø2                    STA CBINV
ØØ57      Ø414     A9  Ø4                        LDA #( BRKE
ØØ58      Ø416     8D  1C  Ø2                    STA CBINV - 1
ØØ59      Ø419     A9  Ø7                        LDA #( EOM
ØØ6Ø      Ø41B     85  7D                        STA VARTAB - 1
ØØ61      Ø41D     A9  6B                        LDA #( EOM
ØØ62      Ø41F     85  7C                        STA VARTAB
ØØ63      Ø421     A9  43                        LDA #'C
ØØ64      Ø423     85  21                        STA TMPC
ØØ65      Ø425     DØ  12                        BNE B3
ØØ66      Ø427     A9  42              BRKE      LDA #'B
ØØ67      Ø429     85  21                        STA TMPC
ØØ68      Ø42B     D8                            CLD
ØØ69      Ø42C     4A                            LSR A
ØØ7Ø      Ø42D     68                            PLA
ØØ71      Ø42E     85  1E                        STA YR
ØØ72      Ø43Ø     68                            PLA
ØØ73      Ø431     85  1D                        STA XR
ØØ74      Ø433     68                            PLA
ØØ75      Ø434     85  1C                        STA ACC
ØØ76      Ø436     68                            PLA
ØØ77      Ø437     85  1B                        STA FLGS
ØØ78      Ø439     68                  B3        PLA
ØØ79      Ø43A     69  FF                        ADC #$FF
ØØ8Ø      Ø43C     85  19                        STA PCL
ØØ81      Ø43E     68                            PLA
ØØ82      Ø43F     69  FF                        ADC #$FF
ØØ83      Ø441     85  1A                        STA PCH
ØØ84      Ø443     BA                            TSX
```

```
0444    86  1F                      STX  SP
0446    58                          CLI
0447    20  F2  04      BS          JSR  CRLF
044A    A6  21                      LDX  TMPC
044C    A9  2A                      LDA  #'*
044E    20  22  06                  JSR  WRTWO
0451    A9  52                      LDA  #'R
0453
0453
0453
0453    85  0D                      STA  BRKF
0455    D0  2B                      BNE  S0
0457    A9  00          START       LDA  #0
0459    85  CA                      STA  TXTPT
045B    85  0D                      STA  BRKF
045D    85  0A                      STA  WRAP
045F    20  F2  04                  JSR  CRLF
0462    A9  2E                      LDA  #'
0464    20  D2  FF                  JSR  WRT

PET  MONITOR  13..1.......PAGE  0003

LINE    LOC         CODE        LINE

0467    A6  20                      LDX  SAVX
0469
0469
0469    E0  02                      CPX  #2
046B    F0  04                      BEQ  ST0
046D    E0  03                      CPX  #3
046F    D0  06                      BNE  ST1
0471    20  3A  06      ST0         JSR  SPACE
0474    20  37  06                  JSR  SPAC 2
0477    20  90  06      ST1         JSR  RDUC
047A
047A    C9  2E                      CMP  #'
047C    F0  F9                      BEQ  ST1
047E    C9  20                      CMP  #$20
0480    F0  F5                      BEQ  ST1
0482    A2  07          S0          LDX  #NCMDS + 1
0484    DD  02  05      S1          CMP  CMDS, X
0487    D0  0F                      BNE  S2
0489    A5  20                      LDA  SAVX
048B    85  0E                      STA  PREVC
048D    86  20                      STX  SAVX
048F    BD  0A  05                  LDA  ADRA, X
0492    48                          PHA
0493    BD  12  05                  LDA  ADRL, X
0496    48                          PHA
0497    60                          RTS
0498    CA              S2          DEX
0499    10  E9                      BPL  S1
```

```
∅49B  A9 3F           ERROPR    LDA # $3F
∅49D  2∅ D2 FF                  JSR WRT
∅4A∅  4C 57 ∅4                  JMP START
∅4A3  38              DCMP      SEC
∅4A4  A5 13                     LDA TMP 2
∅4A6  E5 11                     SBC TMP ∅
∅4A8  85 ∅B                     STA DIFF
∅4AA  A5 14                     LDA TMP 2+1
∅4AC  E5 12                     SBC TMPO + 1
∅4AE  A8                        TAY
∅4AF  ∅5 ∅B                     ORA DIFF
∅4B1  6∅                        RTS
∅4B2  A5 11           PUTP      LDA TMP∅
∅4B4  85 19                     STA PCL
∅4B6  A5 12                     LDA TMP∅ + 1
∅4B8  85 1A                     STA PCH
∅4BA  6∅                        RTS
∅4BB
∅4BB                  DISPLAY MEM SUBR.  SET AR=NUMBER
∅4BB                  OF MEMORY BYTES DISPLAYED
∅4BB                  TMP∅=ADR OF MEM DISPLAYED
∅4BB
∅4BB  85 21           DM        STA TMPC
∅4BD  A∅ ∅∅                     LDY #∅
∅4BF  2∅ 3A ∅6        DM1       JSR SPACE
∅4C2  B1 11                     LDA (TMP∅), Y
∅4C4  2∅ 13 ∅6                  JSR WROB
```

PET MONITOR 13..1..........PAGE ∅∅∅4

```
LINE   LOC    CODE                LINE

∅4C7  2∅ F7 ∅4                  JSR INCTMP
∅4CA  C6 21                     DEC TMPC
∅4CC  D∅ F1                     BNE DM1
∅4CE  6∅                        RTS
∅4CF
∅4CF                  READ AND STORE BYTE.
∅4CF                  NO STORE IF SPACE OR RCNT = ∅
∅4CF
∅4CF  2∅ 5E ∅6        BYTE      JSR RDOB
∅4D2  9∅ ∅                      BCC BY3
∅4D4  A2 ∅∅                     LDX #∅
∅4D6  81 11                     STA (TMPO, X)
∅4D8  C1 11                     CMP (TMPO, X)
∅4DA  F∅ ∅5                     BEQ BY3
∅4DC  68                        PLA
∅4DD  68                        PLA
∅4DE  4C 9B ∅4                  JMP ERROPR
∅4E1  2∅ F7 ∅4        BY3       JSR INCTMP
∅4E4  C6 21                     DEC RCNT
∅4E6  6∅                        RTS
```

```
/Ø179   Ø4E7   A9  1B        SETR        LDA #FLGS
Ø18Ø    Ø4E9   85  11                    STA TMPØ
Ø181    Ø4EB   A9  ØØ                    LDA #Ø
Ø182    Ø4ED   85  12                    STA TMPØ - 1
Ø183    Ø4EF   A9  Ø5                    LDA #5
Ø184    Ø4F1   6Ø                        RTS
Ø185    Ø4F2   A9  ØD        CRLF        LDA #$ D
Ø186    Ø4F4   4C  D2 FF                 JMP WRT
Ø187    Ø4F7
Ø188    Ø4F7                 INCREMENT (TMPØ, TMPØ - 1) BY 1
Ø189    Ø4F7
Ø19Ø    Ø4F7   E6  11        INCTMP      INC TMPØ
Ø191    Ø4F9   DØ  Ø6                    BNE SETWR
Ø192    Ø4FB   E6  12                    INC TMPØ - 1
Ø193    Ø4FD   DØ  Ø2                    BNE SETWR
Ø194    Ø4FF   E6  ØA                    INC WRAP
Ø195    Ø4Ø1
Ø196    Ø5Ø1   6Ø            SETWR       RTS
```

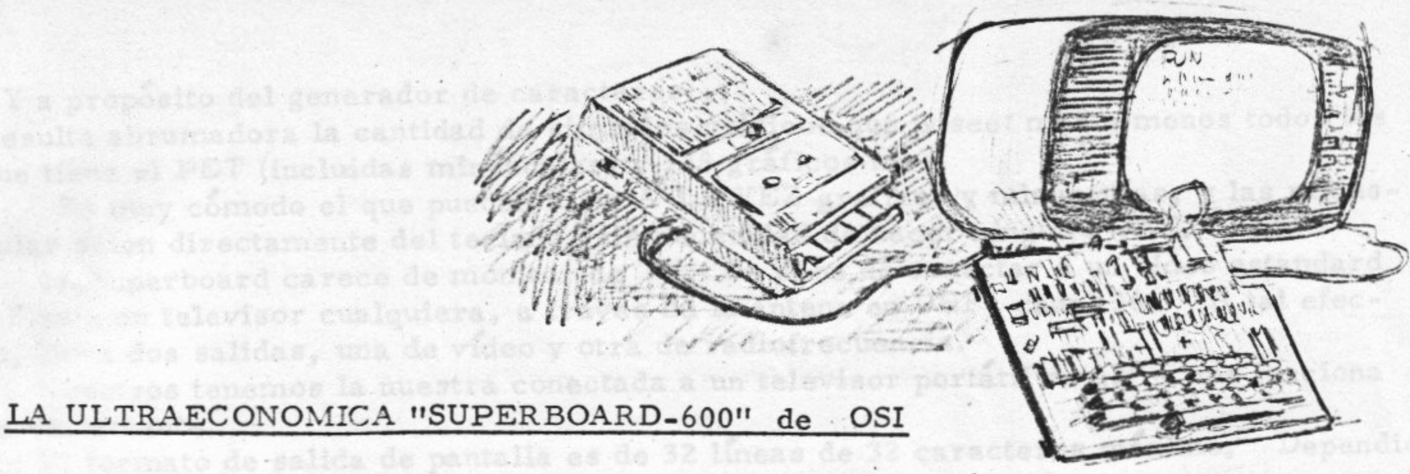
PET MONITOR 13.. 1.......... PAGE ØØØ5

```
LINE    LOC                  CODE        LINE

Ø198    Ø5Ø2   3A            CMDS        .BYTE ':'
Ø199    Ø5Ø3   3B                        .BYTE ';'
Ø2ØØ    Ø5Ø4   52                        .BYTE 'R'
Ø2Ø1    Ø5Ø5   4D                        .BYTE 'M'
Ø2Ø2    Ø5Ø6   47                        .BYTE 'G'
Ø2Ø3    Ø5Ø7   58                        .BYTE 'X'
Ø2Ø4    Ø5Ø8   4C                        .BYTE 'L'
Ø2Ø5    Ø5Ø9   53                        .BYTE 'S'
Ø2Ø5    Ø5ØA   Ø5            ADRH        .BYT )ZZ1
Ø2Ø7    Ø5ØB   Ø5                        .BYT )ZZ2
Ø2Ø8    Ø5ØC   Ø5                        .BYT )ZZ3
Ø2Ø9    Ø5ØD   Ø5                        .BYT )ZZ4
Ø21Ø    Ø5ØE   Ø5                        .BYT )ZZ5
Ø211    Ø5ØF   Ø5                        .BYT )ZZ6
Ø212    Ø5 Ø   Ø6                        .BYT )ZZ7
Ø213    Ø511   Ø6                        .BYT )ZZØ
Ø214    Ø512   C1            ADRL        .BYT )ZZ1
Ø215    Ø513   B1                        .BYT )ZZ2
Ø216    Ø514   2C                        .BYT )ZZ3
Ø217    Ø515   5E                        .BYT )ZZ4
Ø218    Ø516   D7                        .BYT )ZZ5
Ø219    Ø517   FD                        .BYT )ZZ6
Ø22Ø    Ø518   9E                        .BYT )ZZ7
Ø221    Ø519   9E                        .BYT )ZZ8
```

PAGE MONITOR 13..1........ .PAGE ØØØ6

```
LINE    LOC                  CODE        LINE
```



LA ULTRAECONOMICA "SUPERBOARD-600" de OSI

Por fin los sueños de muchos aficionados a la microinformática se hacen realidad!! Los sueños de aquellos que sueñan tener un computador en casa...

Y no es que hasta la fecha fueran imposibles tales sueños, pero, reconzcamoslo, no todo el mundo tiene esas ciento cuarenta y pico mil pesetas que se necesitan para comprar un PET o un TRS-80. Ni mucho menos las alrededor de doscientas mil del APPLE II...

Resulta que ahora nos llevamos la agradable sorpresa:
NOS HAN TRAIDO UN COMPUTADOR CON LAS POSIBILIDADES DE LOS MENCIONADOS Y COMO A UN TERCIO DEL PRECIO!!

...Efectivamente, una casa, que por cierto no nos autoriza aún a decir su nombre (esas manías de los comerciales que no entenderemos nunca los técnicos...), nos ha vendido una de las primeras Superboards que han entrado en España; nos han asegurado que muy pronto tendran muchas mas para la venta, si bien no nos han querido decir donde ni como las piensan vender... (insistimos en lo de último paréntesis).

VALE! dejemosnos de disquisiciones y veamos lo que nos da la Superboard 600 de marras...

En los diez días que llevamos "metiendole mano" a fondo y jugando con ella, llegamos a las conclusiones que siguen:

MODELO: SUPERBOARD 600 (El que va sin caja)

HARDWARE:

TECLADO

El teclado esta directamente montado sobre la placa de circuito impreso, se trata de un teclado estandard de ordenador y GRANDE (Señor!! lo que se agradece un teclado grande como este...)

Notamos a faltar la presencia de un teclado numérico independiente. No obstante no ès grave, resulta bastante fácil enchufarle uno de los de calculadora, además es cuestión de costumbre: se puede trabajar muy bien con las teclas numericas situadas encima de las alfabéticas...

Nos ha gustadó mucho el que se trate de un teclado programable a través del BASIC (Polled) y la facilidad con que se programa.

El teclado se halla situado en la posición de memoria hexadecimal DF00

PANTALLA

Al igual que otros computadores, consta de 1K de memoria y se halla situado entre las posiciones $D000 - D3FF. Cualquier Byete POKAdo entre estas posiciones aparece inmediatamente en la pantalla convenientemente decodificado por el generador de caracteres.

Y a propósito del generador de caracteres...
Resulta abrumadora la cantidad de simbolos gráficos que posee: mas o menos todos los que tiene el PET (incluidas minúsculas) y 128 gráficos mas.

Es muy cómodo el que puedan salir A LA VEZ graficos y minúsculas, y las minúsculas salen directamente del teclado, sin necesidad de hacer ningun POKE.

La Superboard carece de monitor de TV. Se debe de conectar a un video estandard o bien a un televisor cualquiera, a traves de la antena en UHF, canal 36. A tal efecto, lleva dos salidas, una de vídeo y otra de radiofrecuencia.

Nosotros tenemos la nuestra conectada a un televisor portátil de 12" y nos funciona perfectamente.

El formato de salida de pantalla es de 32 líneas de 32 caracteres máximo. Dependiendo del monitor utilizado, se tendran otras salidas distintas pero siempre con menos capacidad.

Cuando lo conectamos al monitor de vídeo del TRS-80 que tenemos aquí, conseguimos que aparezcan 20 líneas de 20 caracteres.

Sin embargo usando el televisor portátil, el formato que nos sale es de 26 líneas de 24 caracteres...

CASSETTE

La Superboard viene con una salida para cassette según el sistema Kansas, trabajando a una velocidad de 300 Baudios.

300 Baudios es una velocidad algo lenta pero queda compensada por su gran fiabilidad.

Se utiliza una ACIA de Motorola como interface, lo cual quiere decir que no hay mayores problemas a la hora de cambiar la velocidad de grabacion/lectura del cassette por tratarse de un elemento asincrono.

De otro lado, esta ACIA permite su adaptación con muy pocos elementos exteriores, como salida para impresora en loop de 20mA o bien en RS-232, todo ello con una sencilla rutina en OS colocada en una zona de la página cero protegida que ya lleva la Superboard para estos casos.

EXPANSIONES FUTURAS

En el centro de la Superboard existe un conector libre de 40 pines preparado para conectar la expansion (placa modelo 610).

Dicha placa 610 de expansión lleva incorporado un reloj en tiempo real, interface para impresora, 24K de memoria, dos controladores para mini-disco y una salida para el bus OSI 48 en el cual se pueden conectar multitud de periféricos o adaptadores a otros tipos de bus.

MONITOR

La Superboard lleva el Monitor de lenguaje de máquina incorporado en ROM.

Dicho Monitor es muy similar al del PET con la ventaja de que no es necesario el cargarlo en cassette cada vez que se necesita; basta con llamarlo mediante el comando "BREAK-M"

BASIC

Se trata de un BASIC típico de 8K en ROM. Realmente poco se puede decir de nuevo del BASIC ya que todos los microcomputadores estan equipados con BASICs escritos por la compañía americana MICROSOFT (PET, TRS-80, APPLE II, ISE, etc.).

La única novedad que le apreciamos es su gran velocidad de ejecución, de alrededor de un 30% más rápido que el PET... Terrible!!!

ESTAMOS APRENDIENDO COSAS NUEVAS DE ESTA PLACA...
EL MES PROXIMO CONTINUAREMOS!